January 2015

A FRAMEWORK FOR OPTIMAL DESIGN OF LOW-POWER FIR FILTERS

Aparajita Banerjee
Purdue University

Graduate School Form 30
(Revised 08/14)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Aparajita Banerjee

Entitled
A Framework for Optimal Design of Low-power Approximate FIR Filters

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

KAUSHIK ROY

ANAND RAGHUNATHAN

VIJAY RAGHUNATHAN

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification/Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): KAUSHIK ROY

Approved by: Michael R. Melloch 05/12/2015

Head of the Department Graduate Program Date

A FRAMEWORK FOR OPTIMAL DESIGN OF LOW-POWER FIR FILTERS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Aparajita Banerjee

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

August 2015

Purdue University

West Lafayette, Indiana

This thesis is dedicated to my family and friends.

ACKNOWLEDGMENTS

I would take this opportunity to sincerely express my gratitude to all the wonderful people that I came across in the last two years. First and foremost, I would like to thank Professor Kaushik Roy for giving me this wonderful opportunity to work with him. His words of encouragement, technical insights and positive outlook influenced me a lot. I would like to thank Professor Anand Raghunathan for all the essential suggestions for my work and also for being on my advisory committee. I would like to thank Professor Vijay Raghunathan for taking out time to be on my committee.
I would also like to thank my fellow researchers for helping me during my research journey. Special thanks to Swagath Venkataramani for helping me out whenever I faced any roadblock and for being so patient and understanding. I would also like to thank Abhronil Sengupta for being an excellent collaborator and a very good friend. I thank all my Nanoelectronics Research Laboratory friends who have helped and supported me during my Master's journey.
And most importantly, I would like to thank my parents for all the motivation and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Banerjee, Aparajita M.S.E.C.E, Purdue University, August 2015. A Framework for Optimal Design of Low-power FIR Filters. Major Professor: Kaushik Roy.

Approximate Computing has emerged as a new low-power design approach for application domains characterized by intrinsic error resilience. Digital Signal Processing (DSP) is one such domain where outputs of acceptable quality can be produced even though the internal computations are carried out in an approximate manner. With the ever increasing need for data rates at lower power usage; the need for improved complexity reduction schemes for DSP systems continues. One of the most widely performed steps in DSP is FIR filtering. FIR filters are preferred due to their linear phase, stability and ease of implementation. In order to build low-power DSP systems; there have been many efforts to reduce the implementation cost of FIR filters with little loss of quality. However most of the implementations fail to guarantee an optimum solution.

In this thesis, we developed a new mathematical framework for designing FIR Filter with reduced complexity so as to improve performance at reduced power. Given some approximate circuits, the idea is to make the design process aware of them. The proposed methodology is formulated as a linear programming problem that minimizes the Chebyshev error. The impact of hardware approximations is quantified as a function of filter coefficients and converted into the frequency domain. This error can be potentially recovered by intelligently designing a framework that solves the filter design problem with “error due to complexity reduction” as an additional constraint. The design methodology generates a set of filter coefficients that satisfies the filter specifications and adheres to a low-power implementation.By doing this in a systematic and guided manner, we can achieve energy efficiency at acceptable quality.

1. INTRODUCTION

The growth of the semiconductor industry has been phenomenal since its inception. The $336 billion revenue industry [1] has primarily been motivated by a prediction 50 years ago: "the number of transistors in an integrated circuit will double approximately every 2 years" popularly known as the Moore's Law. The law deals with a steady rate of miniaturization in technology. The capabilities of of many digital devices such as memory capacity, microprocessor speed, portable devices are strongly linked to Moore's Law. The digital industry is driven by a simple belief: smaller, faster and cheaper.

However, as device sizes are nearing the size of atoms and integration densities are huge; power barriers and design unpredictability have emerged as new challenges in the semiconductor industry. There is a growing need for an alternate design approach for building efficient systems.

1.1 Trends and Challenges in modern semiconductor industry

The age of miniaturization has been driven by novelties in computer architecture, circuit design and fabrication process. However, the primary force behind this growth has been technology scaling. CMOS scaling improves performance by making transistors faster, leads to higher transistor density in integrated circuits (IC) and reduces the power consumed per transistor [2]. Transistor shrinkage along with interconnect and supply voltage scaling have enabled microprocessors with improved computing performance at lower power envelope. Since the number of chips manufactured per wafer has increased, the die cost has reduced significantly. Scaling has allowed system designers to bring more functionality into computing platforms of various form factors. Figure 1.1 shows how the number of computing devices has grown with time as CMOS technology scaled.

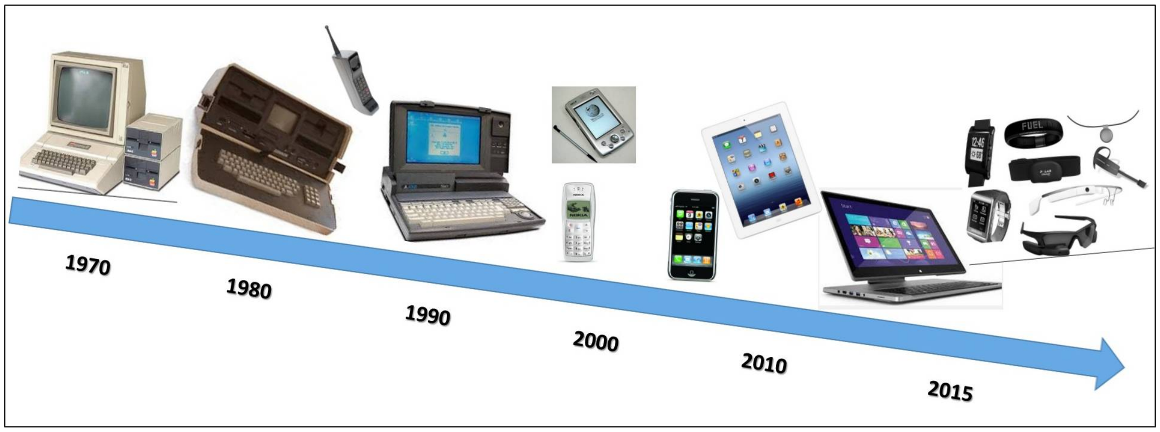


Fig. 1.1. The Evolution of Computing Devices

However recently, aggressive scaling and higher integration density have led to new challenges in the semiconductor industry in terms of power and fabrication limits [3], [4]. As scaling moves into deep-nanometer regime, the benefits are diminishing due to increased power dissipation and design unpredictability.

1.1.1 Power Dissipation

With the growth in the number of smaller form factor portable products, the challenge in recent years has been to sustain performance at lower power envelopes. The demand for high quality applications makes the process of limiting the power consumption extremely difficult. The active power (also called the dynamic power) consumption ($P_{active} = CV^2f$) increases as the operating frequency and the number of transistors on a chip increases. Thus the increased features/functionality comes at the cost of higher power consumption.

A new culprit of high power dissipation in today's chips is the leakage power which grows exponentially [5] with every new technology node. Leakage power is the unwanted idle state power dissipation which results due to the subthreshold current in the transistor channel even when the transistor is off. With continued scaling in the deep nanometer regime, the leakage power becomes comparable to the dynamic power. In Figure 1.2 we observe that the active power have grown linearly with advancing technology nodes whereas the leakage power has shown an exponential growth which is a major concern for the semiconductor industry today.

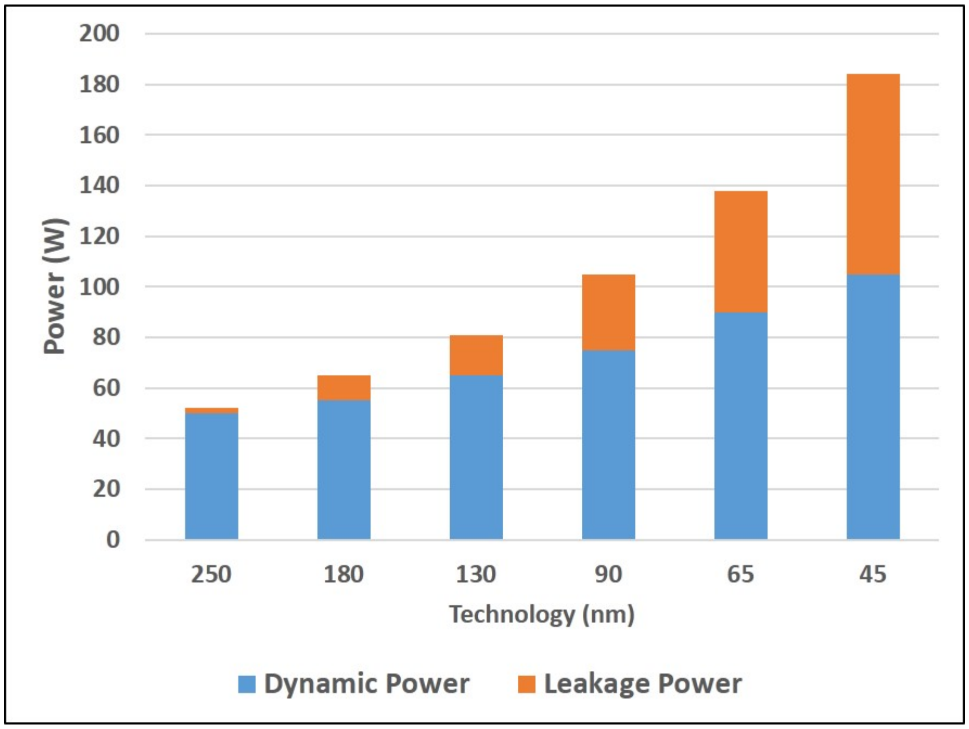


Fig. 1.2. The Trend in Leakage Power with advancing technology node

1.1.2 Process Variations

As scaling continues, the design of high performance circuits have been complicated further by process variations. The control of process variations and fabrication uncertainties is becoming harder with each technology generation. Process variations impact device paramters such as device length L, oxide thickness t_{ox}, threshold voltage V_{th} and mobility μ. Some variations have the same parameter value across the die but differs die to die (*inter-die variations*) whereas some variation affects causes different transistors on the same die to have different parameter values (*intra-die variations*). Process variations introduces unpredictability in the design process and is extremely difficult to predict. It impacts the performance of scaled devices as the can cause a spatial variations in the threshold voltage of the transistors. The issue of process variations has a detrimental effect on the yield rate which in turn affects the economics of the fabrication companies.

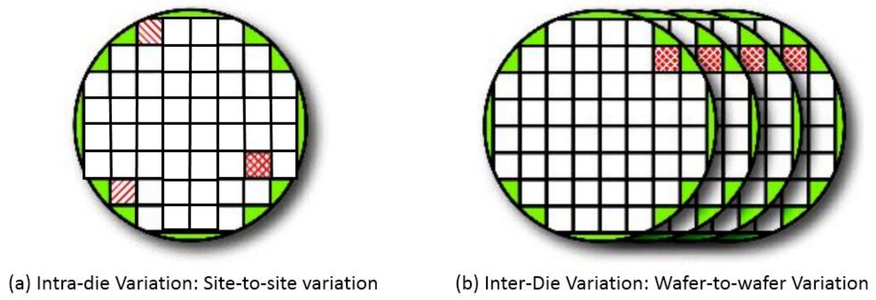

(a) Intra-die Variation: Site-to-site variation
(b) Inter-Die Variation: Wafer-to-wafer Variation

Fig. 1.3. Process Variation: (a) Intra-die Variation, (b) Inter-Die Variation E

These challenges lead to the realization that the benefits from CMOS scaling cannot continue forever.

1.2 Digital Signal Processing Applications

The advent in VLSI has to led to evolution of systems with high throughput capabilities. The domain of Digital Signal Processing (DSP) has been highly influenced by these developments where high throughput is desirable. DSP applications such as image and video processing, speech recognition, digital communication are becoming increasingly popular. Today, we have DSP processors capable of carrying out highly complex computation at speeds favoring real-time operation. In the high performance systems, complex DSP applications such as bio-informatics, facial recognition, business analysis have unprecedented computing needs. This leads to huge amount of power dissipation in servers and mainframe computers. Moreover, installing and maintaining cooling fans and heat sinks adds to the implementation cost of such systems.

One of the fastest growing computing platforms for DSP workloads is the portable form. The workload for these devices have shifted from simple voice/text applications to multimedia services such as web browsing, image processing and accessing videos. The growing popularity for these small, smart and multi-functional devices has led to high demand for better quality in these multimedia services. To fulfill these demands, computing systems should be able to transmit and process data at high data rate. This leads to high power consumption. Providing high performance at low power dissipation has become a major challenge in these portable devices.

Due to the reasons mentioned above, low power design techniques are required to be adapted for DSP systems on portable and non-portable platforms. Clearly, there is a need for energy-efficient implementations for DSP systems.

All these DSP applications have a very interesting characteristic which may help design efficient systems. They are inherently resilient to inaccuracies in their underlying computations. These applications can produce acceptable outputs even in the presence of imprecise input data, noisy communication channels or incorrect intermediate computations without using any error correction scheme. The notion of 100% accurate output becomes unnecessary since the performance requirement is limited by human perception of quality. These applications usually have a statistical computation model (which means the errors could be averaged down) and the algorithms process data in an iterative, successive tuning manner. The figure below lists some of the popular error resilient applications.

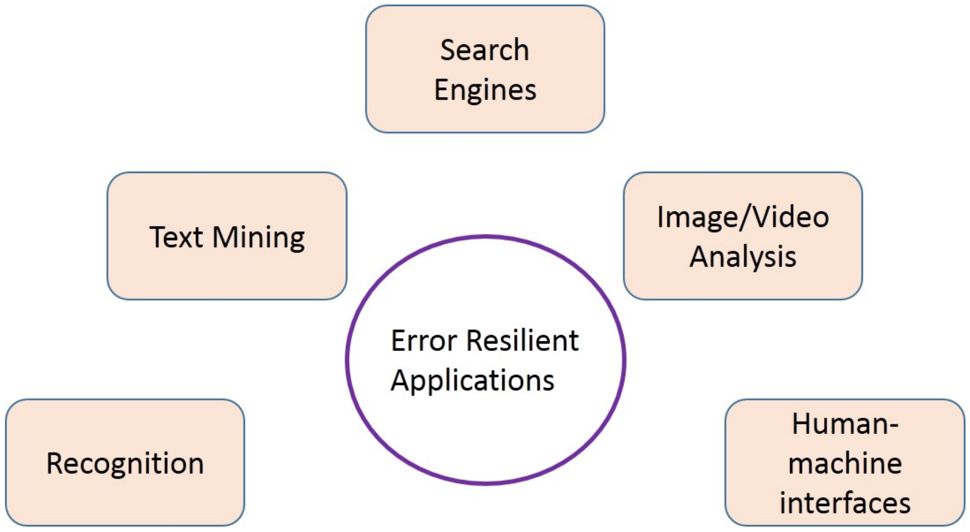


Fig. 1.4. Error Resilient DSP applications E

If we look at the traditional design implementations of such applications; the algorithms focus on "good-enough" user experience and not on exact numerical computation. Their output is defined on a continuous scale of quality. However, as the design moves from software to architecture to circuits, the notion of quality-bound output changes to exactness. At logic and circuits, the focus is on Boolean Equivalence which quantizes quality as either correct or incorrect. From a holistic perspective, it seems to be an unnecessary effort for such computing platforms since it leads to significant efficiency.

1.3 Approximate Computing

With diminishing benefits of scaling and increased computing requirements of the applications shown in Figure 1.4, we need to look for alternate way of designing such systems. There is a need for identifying new source of efficiency across the computing stack to extend the capabilities of future computing platforms such as recognition, associative computing, vision, web search and many more.

Approximate Computing is a design method that [6], [7] takes advantage of the flexibility error resilience applications provide and implements highly optimized hardware for such applications. It is about designing computing platforms that can modulate their efforts by approximating selected computations based on the context to produce just good-enough results. The basic idea is to allow approximations across different layers of abstraction as long as the output quality is acceptable. Approximate computing helps remove the unimportant computations which potentially has minimum effect on application quality but requires significant circuitry and effort just for ensuring Boolean Equivalence. It systematically exploits the Q-E tradeoff at all layers.The design methods under this paradigm are driven by the goal to achieve energy benefits for an acceptable quality loss.

Approximate computing is not a new idea. Significance driven computations [8], accuracy guaranteed bit width optimization [9], heuristics algorithms used in synthesis and placement tools are some of the examples. In all these examples, the insignificant computation were removed by truncating or terminating when a certain desired quality was achieved.

1.4 Thesis Outline

1.4.1 Thesis Objective

The primary objective of this thesis is to achieve superior energy vs quality tradeoff for error resilience DSP applications. As discussed earlier, such applications have the ability to produce good enough results even in the presence of some inaccurate computations. The potential of approximate computing design technique was explored for the domain of DSP specifically in FIR filtering. The objective was to exploit the error resilience nature of the traditional filter design process and design approximate filters systematically in the presence of approximate hardware. We propose a new design framework for FIR filters which can be implemented efficiently in the presence of various approximation schemes. The novelty lies in capturing the errors due to hardware approximations effectively and making the filter design process aware of this design space. The optimization problem is constrained in the presence of these hardware errors and the optimizer modulates the filter parameters, ie, the filter coefficients to reduce the filter error.

1.4.2 Organization

In this thesis, we focus on applying the techniques of approximate computing to the domain of Digital Signal Processing (DSP).The remaining part of the thesis is organized as follows.

Chapter 2 talks about the fundamentals of DSP and its relevance. A brief background of the mathematics of FIR filter design and the implementation structures of these filters are discussed. The chapter concludes with a discussion of the previous efforts of designing energy efficient FIR filters.

Chapter 3 defines the main objective and contributions of this thesis. A new mathematical framework for integrating hardware approximations with the traditional filter design process has been described in detail. Using this framework, an algorithm has been proposed to systematically design and implement energy-efficient FIR Filters.

Chapter 4 examines the results of under various filter specifications. It shows how the Energy vs Quality curve compares to previous attempts of low-power design and shows the potential of this algorithm by comparing the magnitude response of some filters.

Finally conclusions are drawn in Chapter 5.

2. DIGITAL SIGNAL PROCESSING: AN OVERVIEW

Digital signal processing forms the backbone of most of the multimedia applications. They are used in applications spanning from audio processing, digital image processing, consumer electronics, biomedical diagnosis to many other domains. These DSP applications results from well advances in digital filtering. With advents in VLSI technologies, digital filters have become more viable and provides solutions which are practically impossible for analog filters.

2.1 Digital Filters

A digital filter is a function which maps a discrete-time signal to a real or complex valued output signal. The objective of the filter is to enhance or diminish some aspect of the input signal. They are generally used for two general tasks: (a) Separation of signals that have been combined, and (b) restoration of signals that have been distorted by noise. Both the task focuses on retrieving the desired signal from the contaminated signal. For example: an audio signal passing through a noisy channel may contain high frequency random noisy signals. A good characteristic digital filter can reproduce the exact audio signal at the output.

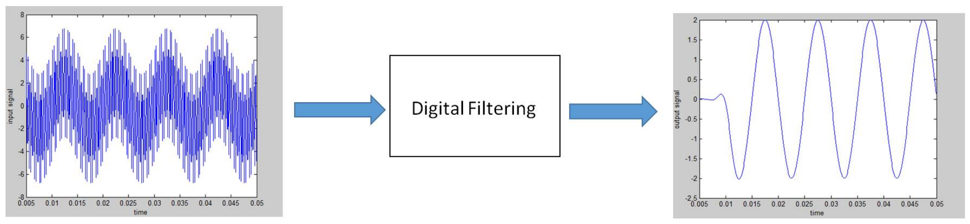


Fig. 2.1. Digital Filtering: To remove noise

The first step of digital filtering is to convert the input analog signal to sampled discrete-time signal. An analog-to-digital converter (ADC) achieves this by quantizing the input signal. The signal is sampled at a frequency fs. For the input analog signal to be properly sampled, it should only contain frequency bands lower than half the sampling frequency fs. If an input signal contains frequency components higher than half the sampling frequency (fs/2), it will cause distortion to the original spectrum. To address this issue, an anti-aliasing filter (AAF) is used before ADC to remove high-frequency components from input frequency spectrum. After sampling, a discrete input signal is passed through the digital filter whose characteristics decide the output sampled signal. To convert the filtered discrete-time signal into an analog signal, a digital-to-analog converter (DAC) is used. The output of DAC may contain some frequency components higher than fs/2 which needs to be eliminated. Again, a low-pass noise filter (NF) with the sampling frequency fs/2 is used to eliminate the higher frequency components. The figure below shows the process of digital filtering.

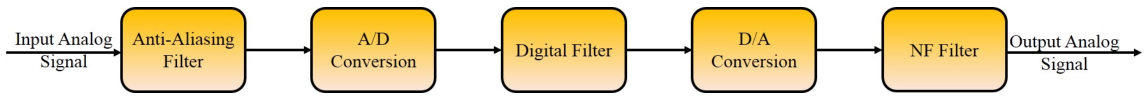


Fig. 2.2. Digital Filtering

2.2 Types of Digital Filters

Digital filters can be broadly classified as: finite impulse response (FIR) and infinite impulse filter (IIR). The classification is based on the impulse response of the filter. FIR filters are characterized by linear phase whereas IIR filters have non-linear phases. The fundamental characteristic of the signal dictates which filter should be used. For example, speech signal can be filtered with non-linear phase characteristics. Since the phase is not that important, it can be processed by FIR or IIR filters. However for signals where phase is important FIR filter is the sole option. Thus, if phase characteristic is of essence, FIR filters are preferred and if only frequency response is of importance, IIR filters are used due to their lower complexities. The basic characteristics of Finite Impulse Response (FIR) filters are:

- linear phase
- stability
- high filter order which leads to complex implementation

The basic characteristics of Infinite Impulse Response (IIR) are:

- non-linear phase
- low filter order makes the implementation simpler

The next section describes FIR filters in detail.

2.3 FIR Filters

Finite-Impulse Response (FIR) filters have been widely used in most of the DSP applications. Figure 2.3 shows how widely FIR filters are used signal processing applications.

For a discrete-time FIR filter of order N, the relationship between the input sampled signal and the output is given by:

$$y[n] = h_0 * x[n] + h_1 * x[n-1] + h_2 * x[n-2] + ... + h_N * x[n-N] \tag{2.1}$$

where h_js represent the filter coefficients , x[n] is an input signal, y[n] is the output signal. Thus, the n^{th} output sample of a FIR filter is the weighted sum of the most recent n samples of the input signal. In mathematical terms, y is the convolution of the input sampled data with the filter coefficients. An N order filter has (N+1) summation terms on the right.

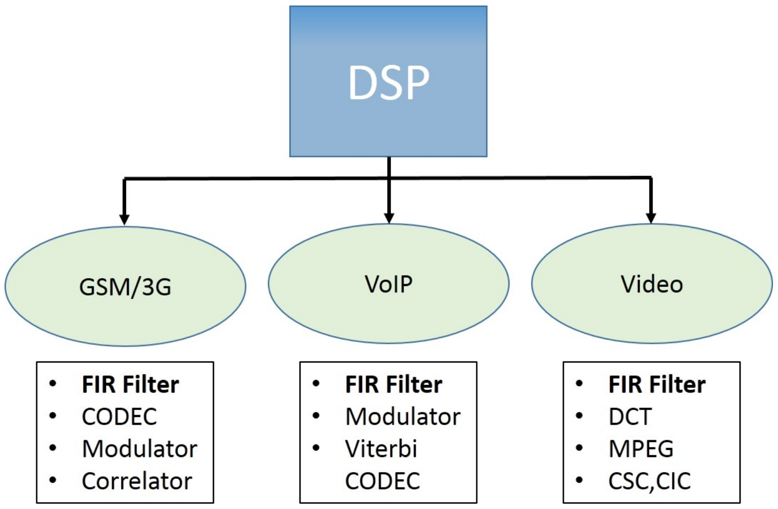


Fig. 2.3. The usage of FIR filters across applications

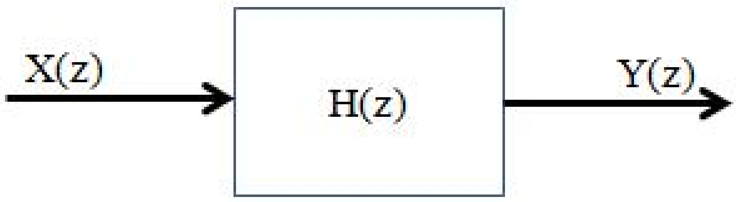


Fig. 2.4. Input-Output Relationship for FIR filter

For a FIR Filter, H(z) represents the frequency response characteristics of the system and is purely determined by the filter coefficients. In the frequency domain, the transfer function of a FIR Filter is given by:

$$H(z) = \Sigma h_i * z^{-i} \tag{2.2}$$

The output in the frequency domain is given as:

$$Y(z) = H(z) * X(z) \tag{2.3}$$

Linear phase characteristics can be achieved in FIR filter only if the impulse response is symmetric or anti-symmetric. It can be expressed as:

h[n] = h[N-n-1] : symmetric impulse response

h[n] = -h[N-n-1] : anti-symmetric impulse response

For designing an FIR Filter, the following parameters characterize the quality:

$$\delta_s : Maximum\ ripples\ in\ the\ stopband \tag{2.4}$$

$$\delta_p : Maximum\ ripples\ in\ the\ passband \tag{2.5}$$

$$\omega_s : normalized\ cutoff\ frequency\ in\ the\ stopband; \tag{2.6}$$

$$\omega_p : normalized\ cutoff\ frequency\ in\ the\ passband; \tag{2.7}$$

2.3.1 FIR Filter Implementation

The most common realization of FIR filters is the Direct Form (DF) structure. This structure uses the filter coefficients directly as the multiplier coefficients.

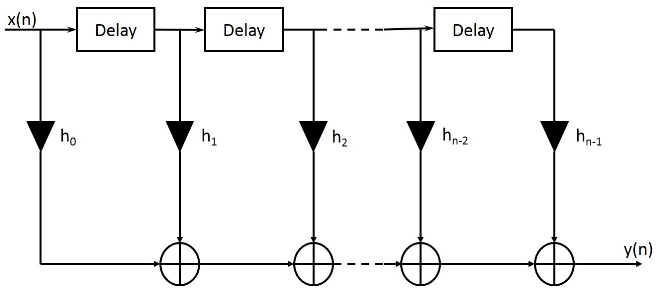


Fig. 2.5. Direct Form Implementation of FIR Filter

In the figure above, we can see that there are three operations implemented in a FIR filter: delay (using a latch), multiplication and addition. The delay units are used to get the past values of x[n]. The number of delay units is same as the order of the filter. The DF requires (N + 1) multipliers, N delay units and N two-input adders for a N order FIR filter. This form is canonical as the number of delays is equal to the order of the transfer function.

Another popular implementation structure is the transpose of the direct form. It is computationally equivalent to the direct form. Only the order of multiplication and delaying is reversed. The figure below shows the transpose form. In this structure, all the multiplications share one input.

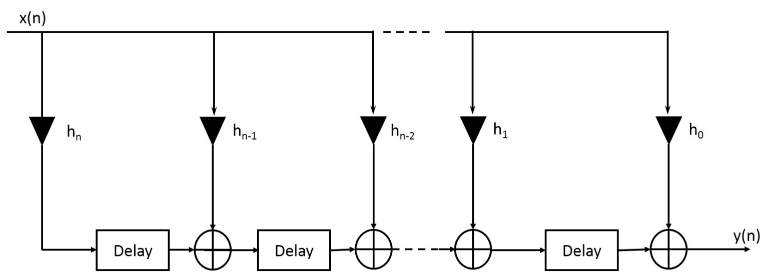


Fig. 2.6. Transpose Direct Form Implementation FIR filter

2.3.2 FIR Filter Design using Optimization Techniques

One common way of designing FIR filters is through ideal filter approximation. The designed filter approximates the ideal characteristic. By increasing the order of the filter, the approximation gets closer to the ideal response.

The design process starts with deciding the filter specifications (setting the parameters in equations 2.4-2.7) of the desirable FIR filter. The specifications are then used to construct a constrained optimization problem.
The traditional filter design method focuses on the objective of reducing the difference between the desired ideal response and the designed response using some optimization scheme. Usually the difference is specified as a weighted function as shown below:

$$E = \{W(\omega)|D(\omega) - H(\omega)|\} \tag{2.8}$$

where

D(ω): Desired frequency response;

H(ω): Frequency response of the filter to be designed;

W(ω): Positive Weighing function to give more importance to the objective function in a certain frequency range;

The transfer function H(ω) is related to the set of filter coefficients as shown in (2.3).

One common approximation measure is given by:

$$minimize||E|| = max\{W(\omega)|D(\omega) - H(\omega)|\} \tag{2.9}$$

This optimization technique is known as the *Chebyshev* criterion which can be stated as: find a set of filter coefficients that minimize the maximum weighted error magnitude.

Another popular optimization technique is minimization of the Least Square Error which is given by:

$$e^2 = \int E^2(\omega)d\omega \tag{2.10}$$

Here, the idea is to minimize the total energy of the error function. However, minimizing the worst case squared error may introduce high error magnitude in some frequency regions in certain formulations.

Frequency domain Chebyshev criterion has been widely used since the Parks McClellan filter Design algorithm. The reason for the popularity of the Chebyshev approximation technique is that it provides a good control of the filter specifications. The error in the filter response is spread evenly across the passband and stopband, which may lead to an optimal solution.

2.4 Low Power Implementation

The central operation in FIR filtering is the multiplication of the input sampled signal by a set of constants (the filter coefficients). This is the most computational intensive step and also the maximum power consuming. The nature of the multiplication, i.e, multiple constant multiplication (MCM) operation in FIR filters can lead to opportunities of power optimization.There has been several research efforts for achieving high-performance and low power implementation of FIR filters.

Some earlier research have investigated the scope of reducing the power dissipation at the algorithm-level or the architecture-level. A well-known low-power scheme is the use of Computational Sharing multiplier (CSHM) algorithm. In CSHM scheme, a small set of bit sequences (called *alphabets*) are selected and stored. These alphabets are then used to generate all the filter coefficients by shift and add operation. CSHM architecture is composed of a precomputer, select units and final adders. The CSHM scheme removes the redundant computations in FIR filtering operation which results in low power and high performance design. Some other techniques such as perturbing the filter coefficients [10], reordering the MAC sequence [11] and use of differential coefficients [12] target reduction in the number of computations while retaining the desired filter characteristics.

One popular approach for designing power efficient FIR filters is multiplierless filters. Simplifying the multiplication by shift-and-add operations, common subexpression sharing and reducing the number of nonzero bits in the binary representation of the coefficients are some of the techniques adopted to reduce the area and power of the implementation. Simpler implementations such as Canonical Signed Digit (CSD) representation [13], Signed-power of two (SPT) representation [14], common subexpression elimination (CSE) techniques [15] and graph synthesis techniques [16] are widely used for designing multiplierless filters. To improve the power and performance further, efficient circuit design techniques [17] have been proposed.

Several research approaches [18], [19] have focused on formulating the objective of minimizing the hardware cost as a discrete optimization problem. The problem is formulated such that, the restrictions on the valid values the filter coefficients become additional constraints to the filter design problem. Various quantization schemes are applied depending on the filter design method and the type of the filter [20]. These techniques introduce errors in the filter's response due to quantization effects.

Many of these approaches start from the optimum filter response and modify the filter coefficients so as to get a low-complexity implementation. The focus is on reducing the implementation cost. The resultant filter is often sub-optimal in quality. Moreover, it is very difficult to predict how much error these schemes could introduce in the filter solution.

The concept of approximate computing has also been used for designing low-power filters. Inaccurate low-power adders and multipliers, designed in [21], [22], have been used in DSP applications and the accuracy vs power trade-off was compared. The inaccurate hardware is used in the implementation of DSP algorithms to analyze the application level quality degradation in comparison to an accurate implementation. These techniques are mostly ad-hoc and the benefits are highly application specific.

3. FIR FILTER DESIGN METHODOLOGY

In this chapter, we propose a novel finite impulse response (FIR) filter design methodology using approximate computing to reduce power consumption and improve performance. We develop a mathematical model to analyze the impact of approximate hardware on the filter design process and systematically design FIR filters in the presence of these approximations.

3.1 The basic Idea

One of the attractive methods of reducing power consumption in DSP applications is complexity reduction. Complexity reduction aims at simplification of certain basic operations. This approach may lower the effective switching capacitance of the underlying hardware. With FIR filtering being an important operation in DSP, low power implementation of these filters can reduce the overall power consumption of DSP systems. Previous efforts on low power FIR filter design have mainly focused on complexity reduction after the filter coefficients are available. In this thesis, we propose a low-power FIR filter design methodology which focuses on designing FIR filters in the presence of hardware approximations. The main idea of this work is to make the traditional filter design process aware of the available hardware (accurate and approximate) by integrating the impact of hardware approximations with it. The expectation is that by modifying the design process, the filter coefficients would get modulated to result in a filter solution that meets the filter specifications in the context of approximate hardware.This approach focuses on an error-aware design.

3.2 Problem Definition

Let us consider the the DF structure of the FIR Filter. The most computational intensive step in FIR filtering is the vector scaling operation where a variable vector is multiplied by a set of constants. With each tap containing a multiplier, there are N + 1 instances of multipliers for a filter of order N. It is observed that a significant amount of power consumption occurs in the multipliers. By reducing the logic complexity in the multipliers, there is an inherent power reduction in the form of lower switched capacitance and leakage due to smaller hardware. However this comes at the cost of imprecision. We explore the opportunities of designing a filter in the presence of simplified multipliers. Given a filter specification, our aim is to develop a systematic methodology to design and implement filters in presence of imprecise and simplified multipliers so as to achieve power and speed improvements without much loss in filter quality. The main challenge is to understand how to modify the traditional filter design process to include the impact of hardware approximations on the overall filter response.
To address this challenge, a 2-step framework has been developed which is described in the next section.

3.3 Proposed Framework

The framework developed answers the following questions.
1. What is the impact of hardware approximations on the filter response?
2. What is the new filter in presence of these approximations?

3.3.1 Step 1: Error Modeling

The logic complexity reduction is applied to multiplication at the bit level by simplifying the multiplier circuit in different ways. Table 3.1 lists three different versions of simplified multipliers. When the hardware logic is approximated, we expect some errors in the output of the multiplier.

Table 3.1: Various Approximate Multipliers Considered

#	Type of Approximation
1	Precision Scaled Multiplier
2	SASIMI based Multiplier [23]
3	IMPACT based Wallace tree Multiplier

The first task is to establish a relationship between the average error observed at the output of the approximated multipliers with the traditional filter design process.

The set of scalars (also known as the filter coefficients) determined by the design process is one of the inputs of the multipliers. The input vectors are the multiplicand and the filter coefficients are the multipliers for the vector scaling operation. The error in the output of the approximated multiplier depends on the input values. By expressing the hardware error as a function of the filter coefficients, we can potentially integrate the impact of hardware approximations on the filter response.

To express the average error of approximate circuits as a function of one of the input nodes; we construct the circuit as shown in figure 3.1. The degree of approximation at the logic level is measured by checking the Boolean difference between the accurate and approximate versions. The circuits listed in Table 3.1 were used as approximate multipliers. The circuits were synthesized in Design Compiler and mapped to IBM 45nm technology. With reference to figure 3.1, for a fixed constant at one input node, we evaluate the error magnitude between the original and the approximate multiplier assuming equi-probable input vectors at the other input node. The average error magnitude is computed by summing the signal probabilities of the output bits of the XOR weighted by their numerical significance.

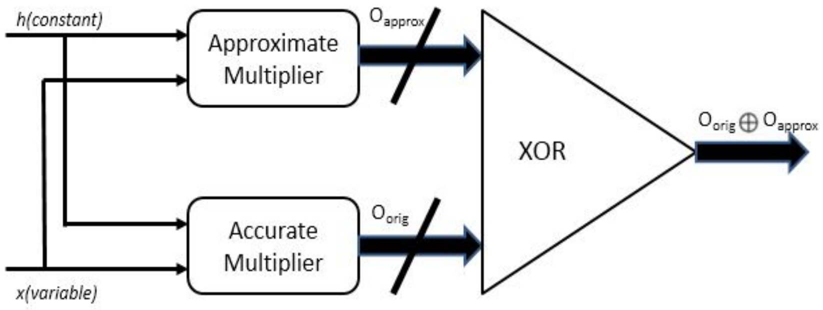


Fig. 3.1. Computing the average Error

In the figure above, h represent the scalar by which the input vectors are multiplied during vector scaling. Therefore, h represent the filter coefficients. By varying the value of h, we get a function of the average error of the approximate multiplier in terms of h. For convenience, we express the average error as a linear function of h. For best fit, the function is made piece-wise linear. Figure 3.2 shows the average error function for a SASAMI based approximate 8-bit multiplier generated by this method.

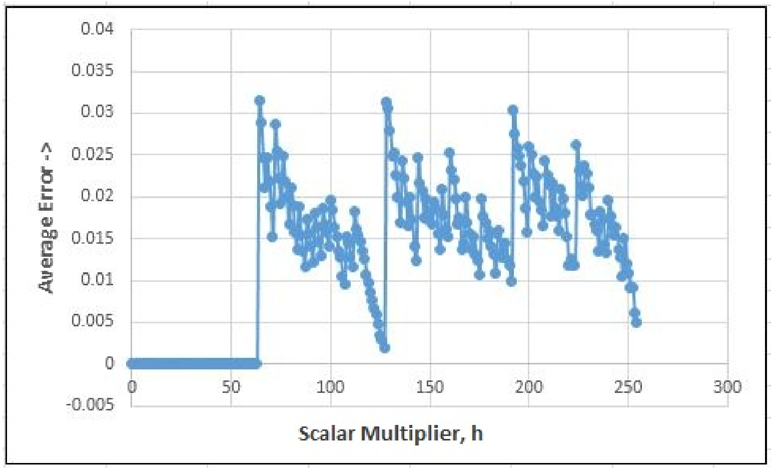


Fig. 3.2. Average Error magnitude as a function of filter coefficient

Table 3.2 lists the various types of approximations considered and the evaluated error function as a linear function of h, where h is an constant multiplier as shown in figure 3.1.

Table 3.2: Various Error Functions

Approximation Type	Function
Type I (Precision Scaled)	f(h) = 0.001*h
Type II (SASIMI based multiplier)	f(h) = 0.0001*(h<16) + 0.0003*(16<h<32) + 0.0005*(32<h<48) + 0.0003*(48<h<64) ...
Type III (IMPACT adder based Multiplier)	f(h) = 0*(h<50) + 0.0003*(50<h<80) + 0.0002*(80<h<250)

Figure 3.3-3.5 shows the average error variation with h for various approximate multipliers. For the precision scaled multipliers, we get a straight linear fit since the variation is small. However for type II and type III, we make it piecewise-linear for better fit. In this step, we successfully modeled the error due to hardware approximations as a linear function of the scalar, h.

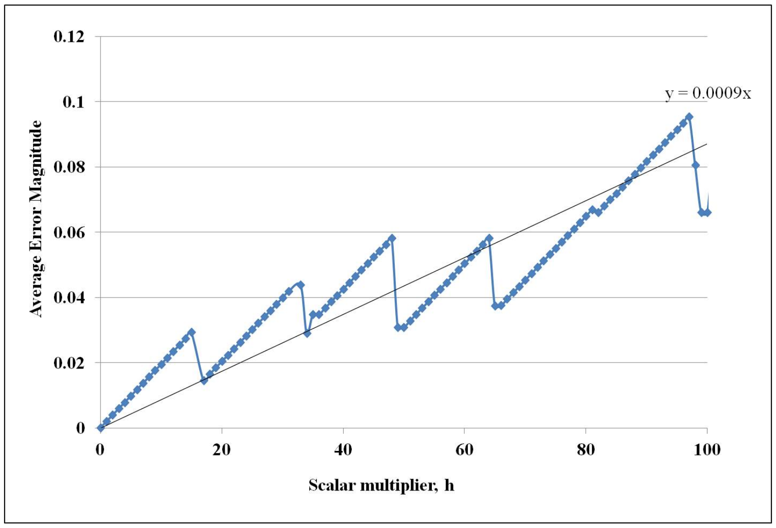


Fig. 3.3. Error Modeling for Precision Scaled Multiplier

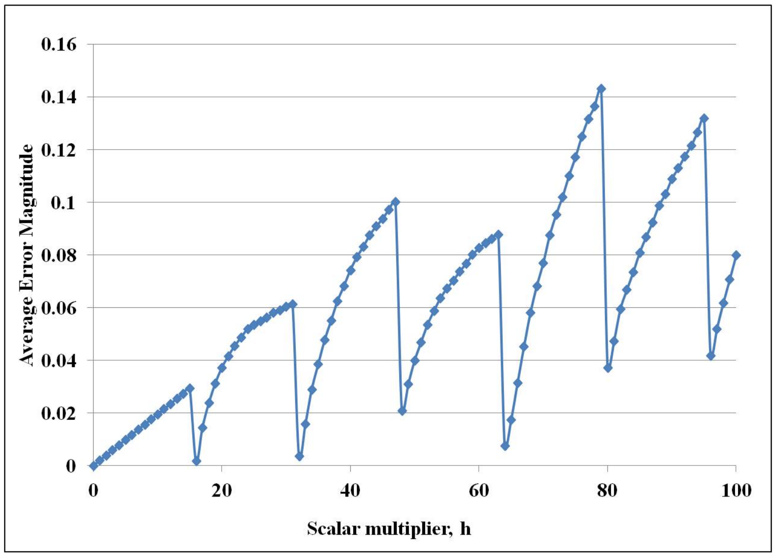


Fig. 3.4. Error Modeling for SASAMI multiplier

The next session talks about how the estimated average error is integrated with the filter design methodology.

3.3.2 Step 2: Formulation of the new Optimization Problem

As discussed in chapter 2, the objective function for FIR filter design using the Chebyshev or min-max criterion is defined as:

$$minimize||E|| = max\{W(\omega)|D(\omega) - H(\omega)|\} \tag{3.1}$$

where D(ω) is the desired frequency response, H(ω) is the frequency response of the designed filter and W(ω) is the weighting function [9]. The optimization problem looks for the optimum set of filter coefficients that minimizes the maximum-weighted absolute error.

As we have seen previously, the transfer function of a direct form FIR Filter of order N can be written as:

$$H(\omega) = \sum_{i=0}^{N} h_i * e^{-ji\omega} \tag{3.2}$$

The h_i in the equation above is same as the x-axis in the error modeling figures. Therefore, using our error characterization model (Table 3.2), the error due to an approximation in the kth multiplier can be expressed as:

$$e_k = a_k * h_k \tag{3.3}$$

where e$_k$ is the error in the multiplication due to an approximation, a$_k$ is a constant by which that particular filter coefficient is multiplied.

In presence of the the error term, the output of the filter becomes:

$$y[n] = \sum_{i=0}^{N} h_i * x[n-i] + \sum_{i=0}^{N} a_i * h_i \tag{3.4}$$

In order to understand the effect of the approximation error on the impulse response of the modified filter, we take the Z-transform of equation 3.4.

$$\begin{aligned} Y(\omega) &= \sum_{n=0}^{\infty} y[n] * z^{-n} \\ &= \sum_{n=0}^{\infty}\sum_{i=0}^{N} h_i * x[N-i] + \sum_{i=0}^{N} a_i * h_i * z^{-n} \\ &= X(\omega) * H(\omega) + \{a_0 + a_1 * z^{-1} + a_2 * z^{-2} + ...\} * \{h_0 + h_1 * z^{-1} + ...\} \end{aligned} \tag{3.5}$$

The impulse response of the modified filter is derived by setting X(ω) = 1 in the equation 3.5. Therefore, our new impulse response become

$$Y(\omega) = H(\omega) + \{a_0 + a_1 * z^{-1} + a_2 * z^{-2} + ...\} * \{h_0 + h_1 * z^{-1} + h_2 * z^{-2} + ...\} \tag{3.6}$$

From the above relationship, it is clear that the error introduced in one of the multiplier affects all the frequency components. The symmetric property of odd/even FIR Filters help simplify the above equation. For our convenience, let's take an even Nth order, symmetric filter. Thus, the number of filter coefficients become N + 1 and they satisfy,

$$h[n] = h[N-n]; 0 < n < N \tag{3.7}$$

So the frequency response can be written as

$$H(\omega) = [h[(N-1)/2] + 2\sum_{i=0}^{(N-1)/2} h[n]cos(((N-1)/2)\omega)]e^{-j\omega((N-1)/2)} \tag{3.8}$$

From (3.6) we observe, in presence of approximations, the filter response includes a secondary term given by:

$$E_a(\omega) = \{a_0 + a_1 * z^{-1} + a_2 * z^{-2} + ...\} * \{h_0 + h_1 * z^{-1} + h_2 * z^{-2} + ...\} \tag{3.9}$$

Combining equations (3.6) and (3.9), we get

$$Y(\omega) = H(\omega) + E_a(\omega); \tag{3.10}$$

where $H(\omega)$ and $\mathrm{E}_a(\omega)$ are given by (3.8) and (3.9) respectively.
Thus, our new filter transfer function can be expressed as a sum of the original frequency response and an error response. In order to get a response as close to the original response as possible in presence of the hardware approximations, we need to modify the problem formulation.
One way to do this is to put an upper limit on the magnitude of the approximation error, E_a.

$$|E_a(\omega)| \leq b \tag{3.11}$$

In order to find the optimum filter coefficients, we combine the initial filter design process (Equation: 3.1) with the additional constraint set by (3.11). The complex optimization problem can now be stated as: *Given an approximation configuration in the form of a vector [$a_0, a_1, a_2...$]; determine the set of filter coefficients that minimizes the Chebyshev error and also satisfies the approximation error constraint (given by 3.11).*

To illustrate the basic principle of the framework, a low pass filter was analyzed in the presence of approximate multipliers and the error recovery scope was analyzed. The error in the filter response was compared with two different filter configurations: Standard with Accurate Realization (SAR) and Standard with Imprecise Realization (SIR). The resultant filter using the proposed method is called Approximation aware Power Efficient Filter (APEF). A brief definition of the three filters is given below:

- Standard with Accurate Realization (SAR): This is the filter designed by solving the original min-max optimization problem and is implemented with accurate multipliers.

- Standard with Imprecise Realization (SIR): In this configuration, the filter coefficients are determined by solving the original min-max criterion. However the implementation includes approximate hardware. The imprecise hardware introduces additional error in the resultant filter response. As the number of inaccurate multipliers increased in the implementation, the error at the filter output also increased.

- Approximation aware Power Efficient Filter (APEF): This filter is designed using our 2-step framework. After the hardware approximations are finalized, the filter design problem is formulated with additional constraints to introduce minimal error due to hardware approximations.

Let's look at an example to analyze the quality of these filters. Example: A half-band FIR Filter of order 24 with passband edge frequency $\omega\text{p} = 0.4\pi$ and stopband frequency $\omega\text{s} = 0.5\pi$. The multipliers were approximated using a precision scaled multiplier.

The error magnitude in filter response for the filters is shown in the figure below. The SAR filter shows a constant error since it's implementation only has accurate multipliers. The SAR filter shows the minimum error possible with the given filter specifications. For the SIR and APEF filters, the error in the filter response increases as the number of approximate multipliers increases in the implementation. However, the increase is much faster when the approximations are applied as an afterthought as seen in SIR filters. For our approximation aware design methodology, the filter coefficients get modified to give a better solution in the presence of imprecise hardware. Clearly, APEF filters have a lower error magnitude even though it has the same degree of hardware approximations as SIR filters.

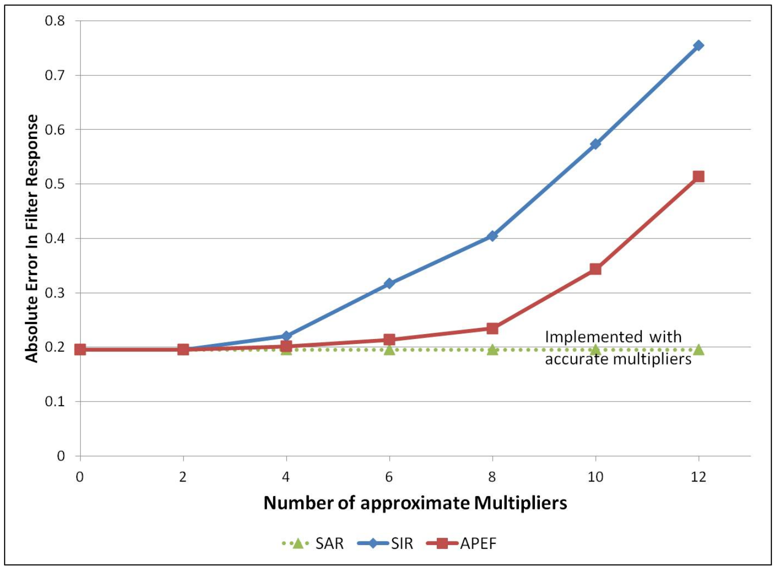


Fig. 3.5. Quality degradation analysis in Filter response

As the degree of approximation increases, the error recovery scope of our framework also rises and thus, the gap between the error response for SIR and APEF increases. Beyond the usage of 8 approximate multipliers; the error in the filter response starts rising steeply. By modifying the filter coefficients, APEF filters can potentially recover more than 40% error and thereby, improve the filter quality. From the figure, we can conclude that our proposed framework results in a solution bounded by SAR and SIR filters.

The proposed method results in both quality and power savings. We compare the implementation cost and quality when 6 instances of accurate multipliers have been substituted by precision scaled multipliers. Table 3.3 lists the power consumption and the area of the three filters. Simpler hardware in the later two implementations result in power and area benefits. However, with approximation aware design approach, the quality of the filter is enhanced at lower power dissipation by modifying the coefficients. Thus, we achieve a superior quality and power product for APEF filters. Table 3.4 shows the coefficient values for the SAR and the APEF filters for the low pass filter example discussed above.

Table 3.3
Power, Area and Quality comparison

Filter Configuration	**Filter Error**	**Power Consumed** (uW)	**Total Area** (μm^2)
SAR filter	0.1953	17.8	18403
SIR filter	0.3269	16.2	17421
APEF filter	0.2132	16.4	17511

Table 3.4
The filter coefficient values for the three filters

Index of taps	**SAR filter**	**SIR filter**	**APEF filter**
$\mathrm{h}_0 = h_{24}$	-0.0217	-0.0217	-0.0217
$\mathrm{h}_1 = h_{23}$	0.0106	0.0106	0.0105
$\mathrm{h}_2 = h_{22}$	0.0265	0.0265	0.0263
$\mathrm{h}_3 = h_{21}$	0.0087	0.0087	0.0086
$\mathrm{h}_4 = h_{20}$	-0.0268	-0.0268	-0.0270
$\mathrm{h}_5 = h_{19}$	-0.0144	-0.0144	-0.0147
$\mathrm{h}_6 = h_{18}$	0.0389	0.0389	0.0386
$\mathrm{h}_7 = h_{17}$	0.0431	0.0431	0.0427
$\mathrm{h}_8 = h_{16}$	-0.0431	-0.0431	-0.0439
$\mathrm{h}_9 = h_{15}$	-0.0904	-0.0904	-0.0908
$\mathrm{h}_{10} = h_{14}$	0.0486	0.0486	0.0481
$\mathrm{h}_{11} = h_{13}$	0.3143	0.3143	0.3138
h_{12}	0.4531	0.4531	0.4496

3.4 The Proposed Algorithm

In this section, we focus on systematic application of our framework for designing FIR filters in the presence of multiple versions of hardware approximations. Given the filter specifications and the various approximation mechanisms, the proposed algorithm analyzes the error recovery scope across all the taps and iterates till the quality constraint is violated. The flowchart below gives a brief overview of the proposed algorithm.

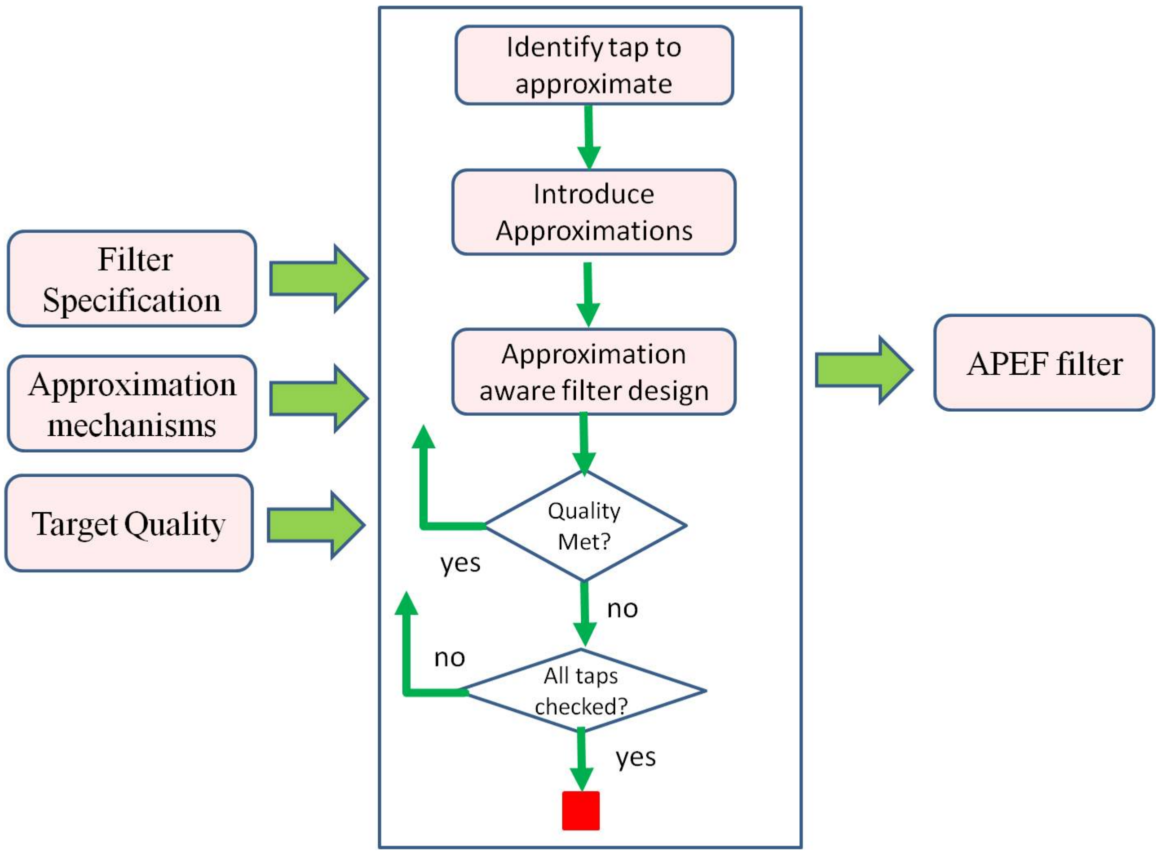


Fig. 3.6. The Proposed Algorithm

The filter characteristics are given as one of the inputs to the algorithm. These include the order of the filter, the passband and stopband frequencies, the desired response, the weighting function W and the number of bits. The order tells us the number of accurate instances of multipliers in the filter. The second input, the approximation mechanisms, lists the type and the version of approximate multipliers. A N^{th} order filter with M approximate versions of the multiplier results in a M*N search space to design an efficient FIR filter. The proposed algorithm iterates over each tap using Gradient Search Method. The model can be divided into the following steps:

- *Rank each configuration*: The error introduced by each approximate version of a multiplier would depend on the filter coefficient since it is one of the inputs. A score based ranking system is chosen to quantize the impact of using different approximation mechanisms at different tap index. The score highlights the amount of energy saved by using the approximate versions divided by the quality loss incurred. For each tap, we have M scores corresponding to each approximate multiplier. For N order filter, M*N scores have to be evaluated. This is a one-time M*N computation. The algorithm starts by chosing the tap with the highest score.

- *Introduce Approximations and design the filter*: After the tap index and the approximation mechanism have been identified (the one with the highest score), use the framework described in Section 3.3 to design the filter so as to generate the APEF filter solution. At each stage, the filter coefficients gets modified to minimize the filter error in presence of hardware approximations.

- *Iterate and backtrack*: For each tap, carry out the previous step in the descending order of the score. After designing, check whether the quality constraints are met and proceed again. In case the quality specifications are violated before all the taps are analyzed, back track to the last "good" configuration and apply the *Gradient Search Method* along a different path.

- *Terminate*: Stop the search when all the taps have been checked. The last configuration which met the quality constraints would be the final optimum APEF filter that conforms to the filter specifications.

By using this algorithm, we can systematically introduce approximations at different tap positions and the our optimization framework modulates the filter coefficients to result in an optimum solution. Both power and quality gets optimized. The simpler hardware results in power savings and the proposed framework leads to error recovery. The combined effect results in an efficient system.
The next section shows how in the presence of various approximation mechanisms, the proposed algorithm can lead to superior energy vs quality trade-off.

4. EXPERIMENTAL METHODOLOGY AND RESULTS

4.1 Experimental Setup

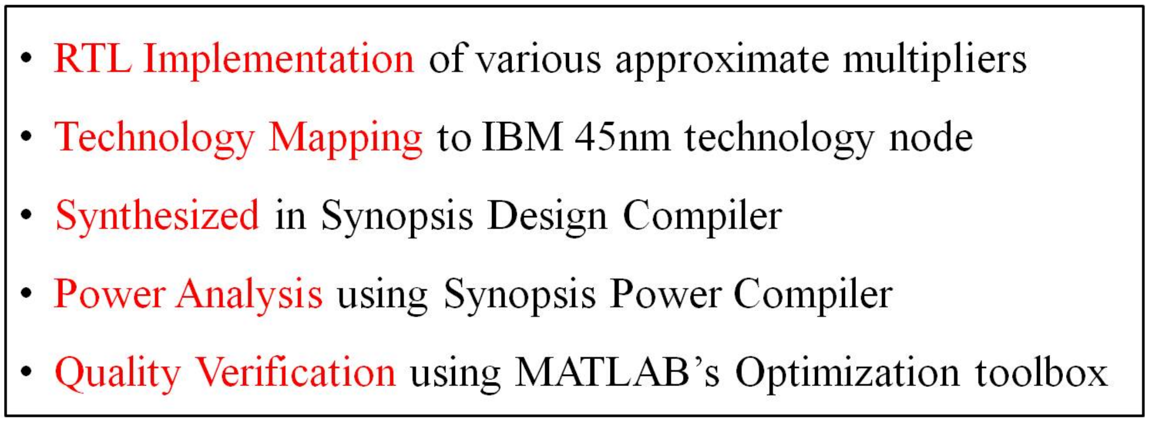
We evaluated the proposed algorithm on different linear filters with different filter characteristics. For modeling the hardware error, all the filters were implemented in the DF structure with delay units, multipliers and adders. Three different types of approximate multipliers were used to demonstrate the potential of the proposed approach as shown in the table below:

Table 4.1
Different Versions of Approximate Multipliers

Type of Approximation	**Number of Approximate Versions**
Precision Scaling	3 (2 bits, 4 bits, 6 bits)
SASIMI based multipliers	2
Impact adder based Wallace tree multiplier	2 (4 LSBs, 8 LSBs)

The second column represents the degree to which each type of approximation is applied to the accurate multiplier. For example: for the precision scaled multipliers, we have 3 different versions where 2,4,6 bits of the input are truncated respectively. With increase in the level of truncation, the power savings per multiplier increases but at the expense of higher average error. Our framework takes as input the filter specifications and the approximation mechanism listed above.

The following diagram shows the various tools used, from error modeling to quality verification.



- RTL Implementation of various approximate multipliers
- Technology Mapping to IBM 45nm technology node
- Synthesized in Synopsis Design Compiler
- Power Analysis using Synopsis Power Compiler
- Quality Verification using MATLAB's Optimization toolbox

Fig. 4.1. The Experimental Setup

First, each version of approximate multiplier is implemented in RTL. Thereafter, it is synthesized and the static signal probability is analyzed. For modeling the approximation error as a function of filter coefficient, the XOR circuit (shown in figure 3.1) is iteratively synthesized and the average error magnitude is computed for each value of the scalar multiplier. Using Synopsis Power Compiler, we record the power consumed in each of the approximate versions.

The nonlinear optimization solver in MATLAB's Optimization toolbox was used for modeling and solving the filter design problem. The *fmincon* function is used to find a constrained minimum of the new objective function.

4.2 Results

The results have been organized in three different sections. The first section compares the energy vs quality trade-off for *SIR* and *APEF* filters. In the second section, the magnitude response for three different filters is presented to compare the quality of APEF filters with SAR and SIR filters for different approximation schemes. The final section tabulates the min-max error for different kinds of filters and compares it with the accurate filter.

4.2.1 Quality-Energy Tradeoff

We compared the quality vs energy tradeoff for SIR and APEF filters on three different examples.

Example 1:

Type of Approximation: *Precision Scaled 8-bit Multiplier*

Type of Filter: *BandPass Filter*

Order of the Filter: 60

Number of approximate multipliers available: 3

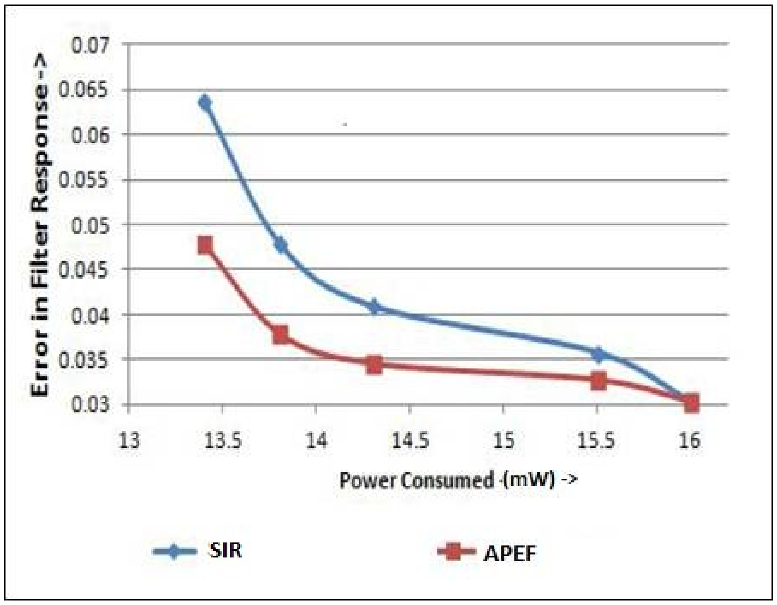


Fig. 4.2. Energy vs Quality tradeoff (Example 1)

When the accurate multipliers are replaced by the approximate version, the quality of the filter degrades. However, the power consumption reduces due to simpler hardware. The blue curve in Figure 4.2 shows the trend when approximations are applied as an afterthought. Designing in the presence of these approximations results in the red curve in the same figure. We observe that for the APEF filter, we get a solution which is better than SIR in terms of power and quality. For the APEF filter, the filter coefficients are modified so as to reduce the filter output error. Therefore, the error in the filter's response reduces and we get a superior Energy vs Quality curve.

Example 2:

Type of Approximation: *SASIMI based 8-bit Multiplier*

Type of Filter: *HighPass Filter*

Order of the Filter: 24

Number of approximate multipliers available: 2

Example 3:

Type of Approximation: *IMPACT based 8-bit Wallace Tree Multiplier*

Type of Filter: *LowPass Filter*

Order of the Filter: 30

Number of approximate multipliers available: 2

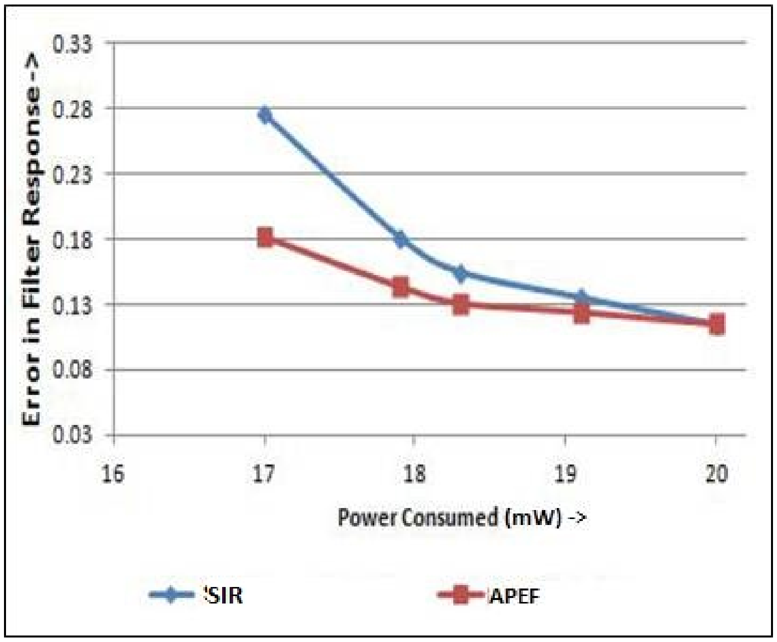


Fig. 4.3. Energy vs Quality trade-off (Example 2)

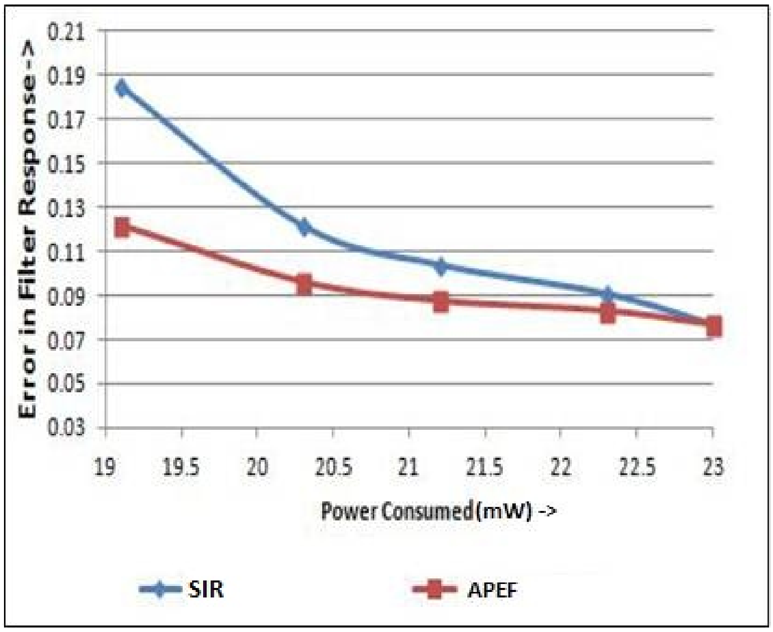


Fig. 4.4. Energy vs Quality trade-off (Example 3)

Figure 4.2 and 4.3 shows the Q-E curve for examples 2 and 3 respectively. As the absolute error rises, the recovery scope increases and we are able to recover more error by using the proposed method. From all the three examples, we can safely conclude that APEF filter results in a better Q-E in comparison to an approximate implementation.

4.2.2 Magnitude Response

In this section, the magnitude response of various linear filters are presented to demonstrate the quality of the filter designed by our methodology.

We observe that the proposed algorithm can significantly reduce the error in filter quality by modulating the filter coefficients. In the first example, the approximations leads to incorrect characteristics at lower frequency ranges. Our method can shift the optimum point even in presence of these approximations to improve the quality of the filter as shown in figure 4.5.

Figure 4.5-4.8 shows the response for the three examples discussed above.

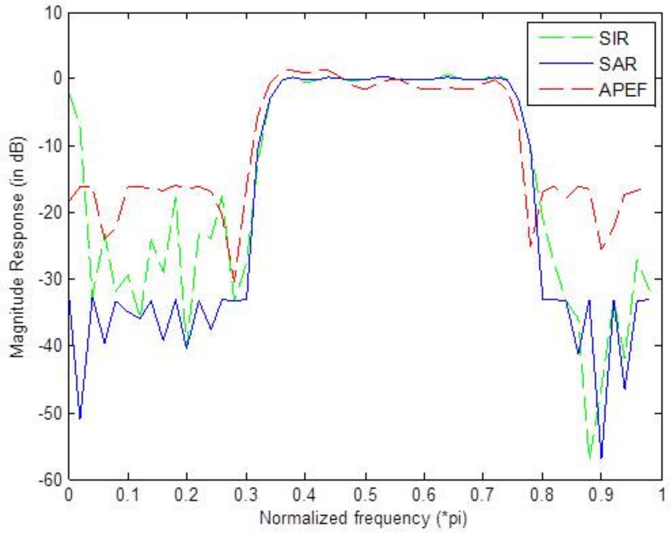


Fig. 4.5. Magnitude Response of N=60 band-pass Filter(Example 1)

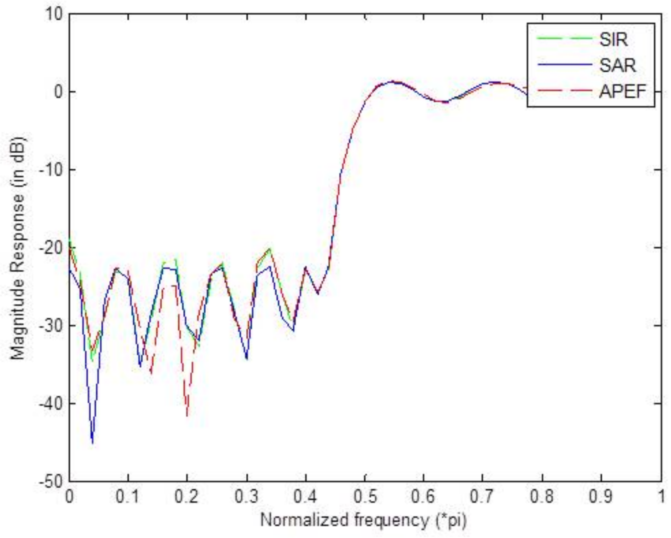


Fig. 4.6. Magnitude Response of N=24 high-pass Filter(Example 2)

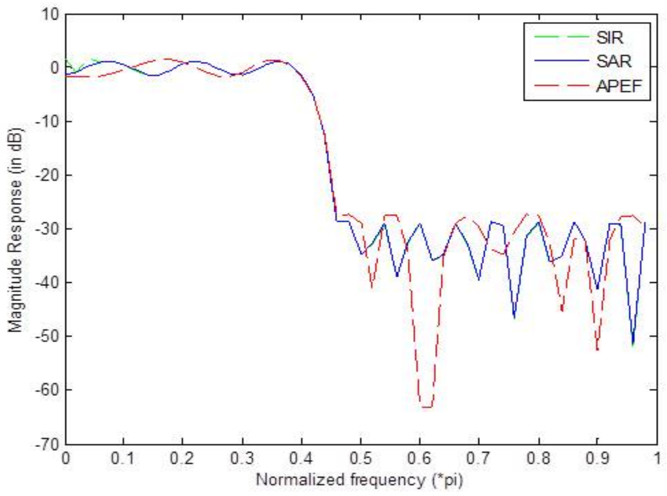


Fig. 4.7. Magnitude Response of N=30 low-pass Filter(Example 3)

4.2.3 Summarizing the results

In this section, we present a table with the error in the objective function for different types of filters under different approximation schemes. Also, different bit widths were used for representation of the coefficient values. The approximation scheme refers to the type of approximate multipliers used. In the error section, the three different configurations are:

SAR: Standard filter with Accurate Realization;

SIR: Standard filter with Imprecise Realization;

APEF: Approximation-aware Power Efficient Filter;

Table 4.2
Filter Response Error for different types of Filters

Filter Specification					Approximation Scheme	Error in the objective function		
Type	Pass	Stop	Order	Bits		SAR	SIR	APEF
Low	0.3	0.4	80	12	Precision Scaled	0.008	0.022	0.0091
High	0.5	0.45	74	12	Precision Scaled	0.003	0.014	0.0058
Band	0.45 0.75	0.35 0.8	64	12	Precision Scaled	0.019	0.054	0.022
Low	0.5	0.58	40	8	SASIMI	0.0146	0.082	0.026
High	0.3	0.25	32	8	SASIMI	0.050	0.120	0.081
Band	0.33 0.66	0.25 0.75	32	8	SASIMI	0.0772	0.194	0.098
Low	0.5	0.65	32	12	IMPACT	0.0071	0.0112	0.0096
High	0.3	0.25	32	12	IMPACT	0.050	0.198	0.083
Band	0.33 0.66	0.20 0.85	50	12	IMPACT	0.0092	0.0154	0.0099

From the table, we make the following observations.

- As the order of the filter increases, the min-max error reduces. This is indepedent of our algorithm.
- As the order of the filter increases, the potential for recovery increases. For the first three rows, we observe the approximate error can be significantly reduced by using the proposed algorithm.
- The percentage error retrieved is high when the degree of approximations is high. This can be observed in the case of a high pass filter with equal specifications approximated by SASIMI and IMPACT based circuits. The approximate error is substantially higher for the IMPACT based substitution. For the APEF filter,

the percentage error retrieval is 58.5% for the IMPACT based approximations whereas the percentage is 32.1% for the SASIMI scheme.

5. CONCLUSION

The support of complex real-time DSP applications on tiny and multi-functional devices has been possible mainly due to advances in the field of VLSI. However, high integration density, large computational requirements and high leakage loss has led to increased power consumption in DSP systems. A number of low-power techniques have been adopted to address the power challenges in such systems.

A widely popular operation in DSP applications is FIR filtering. In order to build low-power DSP systems, there has been many efforts of designing efficient FIR filters. To reduce power consumption in filters while enhancing performance, intelligent trade-offs need to be done between circuit metrics: power and speed and system level metrics: quality and robustness.

In this thesis, we presented a novel framework to design FIR filters using the design techniques of approximate computing. The central idea is to develop a filter design methodology that can generate an optimum filter in the presence of hardware approximations. The impact of the approximations in the hardware were successfully integrated with the design process. The new design methodology generates a set of filter coefficients that satisfies the filter specifications and at the same time conform to a low-power implementation. The proposed algorithm resulted in a systematic design of power-efficient FIR filters.

The algorithm was tested on different kinds of filters when different approximate arithmetic circuits were used. On an average, it resulted in more than 30%power savings in comparison to the standard FIR filter. When compared with a SIR filter; the error in the filter response of APEF was almost 40% lower. The merit of the proposed algorithm is that it takes advantage of the information available at the circuit level and provides feedback to the design stage to result in a superior quality vs error tradeoff.

LIST OF REFERENCES

LIST OF REFERENCES

[1] S. I. Association, “Global semiconductor industry posts record sales in 2014,” 2014.

[2] R. Gonzalez, B. Gordon, and M. Holowitz, “Supply and threshold voltage scaling for low-power cmos,” *Magnetics, IEEE Transactions on*, vol. 32, no. 8, pp. 1210–1216, 1997.

[3] S. Borkar, “Design challenges of technology scaling,” *IEEE Micro*, vol. 19, no. 4, pp. 23–29, 1999.

[4] S. Borkar, “Designing reliable systems from unreliable components: the challenges of transistor variability and degradation,” *IEEE Micro*, vol. 25, no. 6, pp. 23–29, 2005.

[5] A. Ferre and J. Figueras, “Characterization of leakage power in CMOS technologies,” in *Proc. IEEE Int. Conf. On Elctronics, Circuits and Systems*, vol. 2, pp. 85–188, 1998.

[6] V. Chippa, “Approximate computing: An integrated hardware approach,” in *Proc. ASILOMAR*, pp. 111–117, 2013.

[7] Z. Vasicek and L. Sekanina, “Evolutionary design of approximate multipliers under different error metrics,” in *Proc. IEEE Int. Symposium on Design and Diagnostics of Electronic Circuits & Systems*, pp. 135–140, 2014.

[8] D. Mohapatra, G. Karakonstantis, and K. Roy, “Significance driven computation: a voltage-scalable, variation-aware, quality-tuning motion estimator,” in *Proc. IEEE/ACM Int. Symposium on Low Power On Elctronics and Design*, pp. 195–200, 2009.

[9] D.-U. Lee and A. Gaffar, “Accuracy-guaranteed bit-width optimization,” *IEEE Trans. Computer-Aided Design Integr. Circuits Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.

[10] M. Mehendale, S. Sherlekar, and G. Venkatesh, “Coefficient optimization for low power realization of fir filters,” in *VLSI Signal Processing, VIII, 1995. IEEE Signal Processing Society [Workshop on]*, pp. 352–361, Oct 1995.

[11] M. Mehendale, S. Sherlekar, and G. Venkatesh, “Techniques for low power realization of fir filters,” in *Design Automation Conference, 1995. Proceedings of the ASP-DAC ’95/CHDL ’95/VLSI ’95., IFIP International Conference on Hardware Description Languages. IFIP International Conference on Very Large Scal*, pp. 447–450, Aug 1995.

[12] N. Sankarayya and K. Roy, "Algorithms for low power and high speed fir filter realization using differential coefficients," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 44, no. 6, pp. 488 – 497, 1997.

[13] R. Hartley, "Optimization of canonic signed digit multipliers for filter design," in *Proc. IEEE Int. Symposium on Circuits and Systems*, pp. 468–473, 2001.

[14] H. Samueli, "An improved search algorithm for the design of multiplierless fir filters with powers-of-two coefficients," *IEEE TCAS*, pp. 1044–1047, 1989.

[15] M. Mehendale, S. D. Sherlekar, and G. Venkatesh, "Synthesis of multiplierless fir filters with minimum number of additions," in *Proc IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 668–671, 1995.

[16] A. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," in *Proc Inst. Elec. Eng. Circuits and Systems*, pp. 407–413, 1994.

[17] J. Park and K. Roy, "High performance and low power fir filter design based on sharing multiplication," in *Proc. IEEE Int. Symposium on Low Power Electronics and Design*, pp. 295–300, 2002.

[18] M. D. Macleod, "Multiplierless FIR filter design algorithms," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 186–189, 2005.

[19] J. Fuchs, "A class of approximate fir low-pass filters," in *Proc. IEEE Int. Symposium on Circuits and Systems*, pp. 81–84, 2001.

[20] M. Potkonjal, M. B. Srivastaca, and A. Chandrakasan, "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Computer-Aided Design Integr. Circuits Systems*, vol. 15, no. 2, pp. 151–165, 1995.

[21] V. Gupta, M. Debabrata, S. P. Park, R. Anand, and K. Roy, "IMPACT: imprecise adders for low-power approximate computing," in *Proc. 17th IEEE/ACM international symposium on Low-power electronics and design*, pp. 409–414, 2009.

[22] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading accuracy for power with an underdesigned multiplier architecture," in *VLSI Design Int. Conference*, pp. 346–351, 2011.

[23] S. Venkataramani, "Substitute-and-simplify: A unified design paradigm for approximate and quality configurable circuits," in *Proc. DATE*, pp. 111–117, 2013.